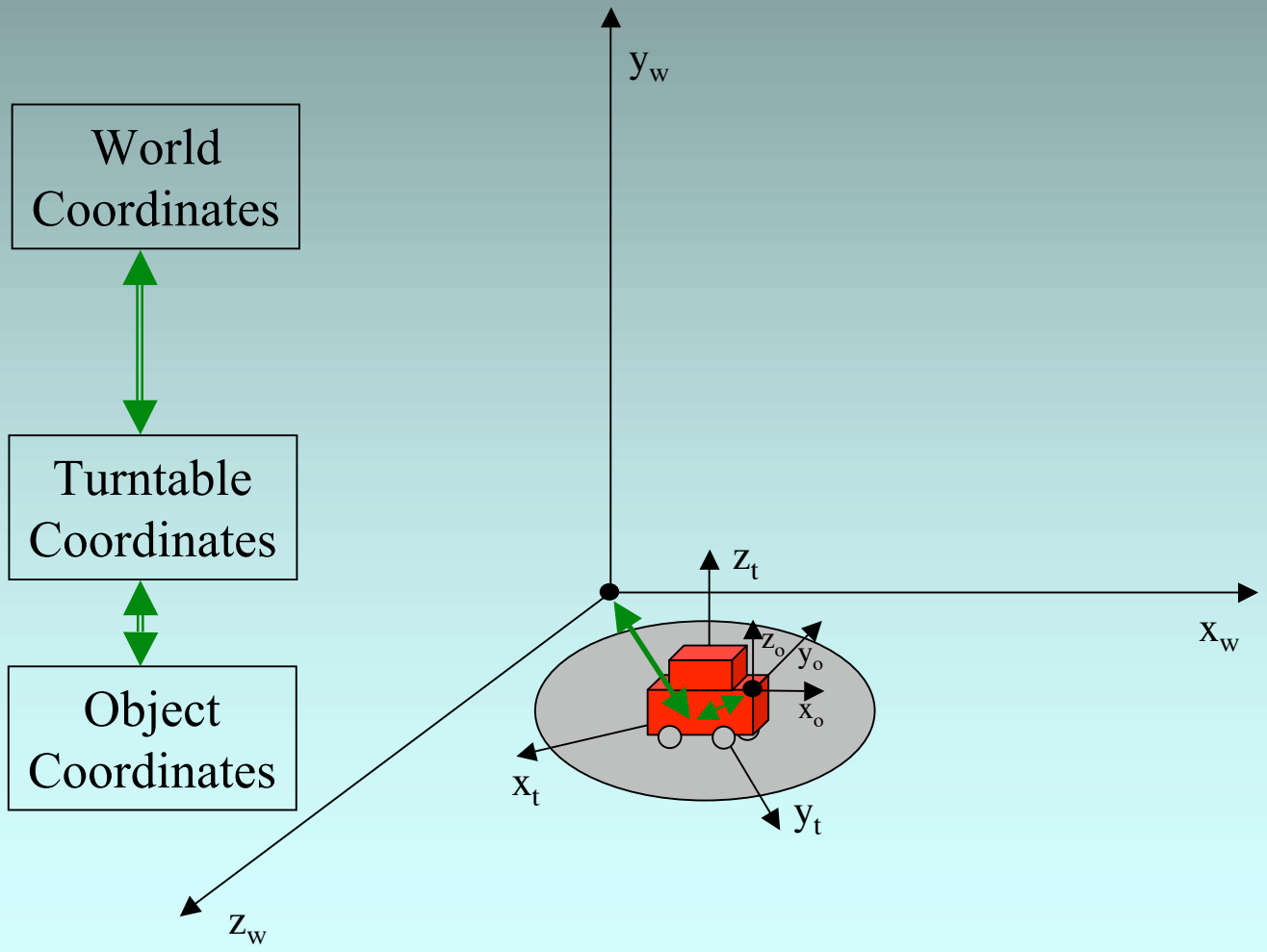
Einführung in die Erweiterte Realität
- 10. Camera Calibration -

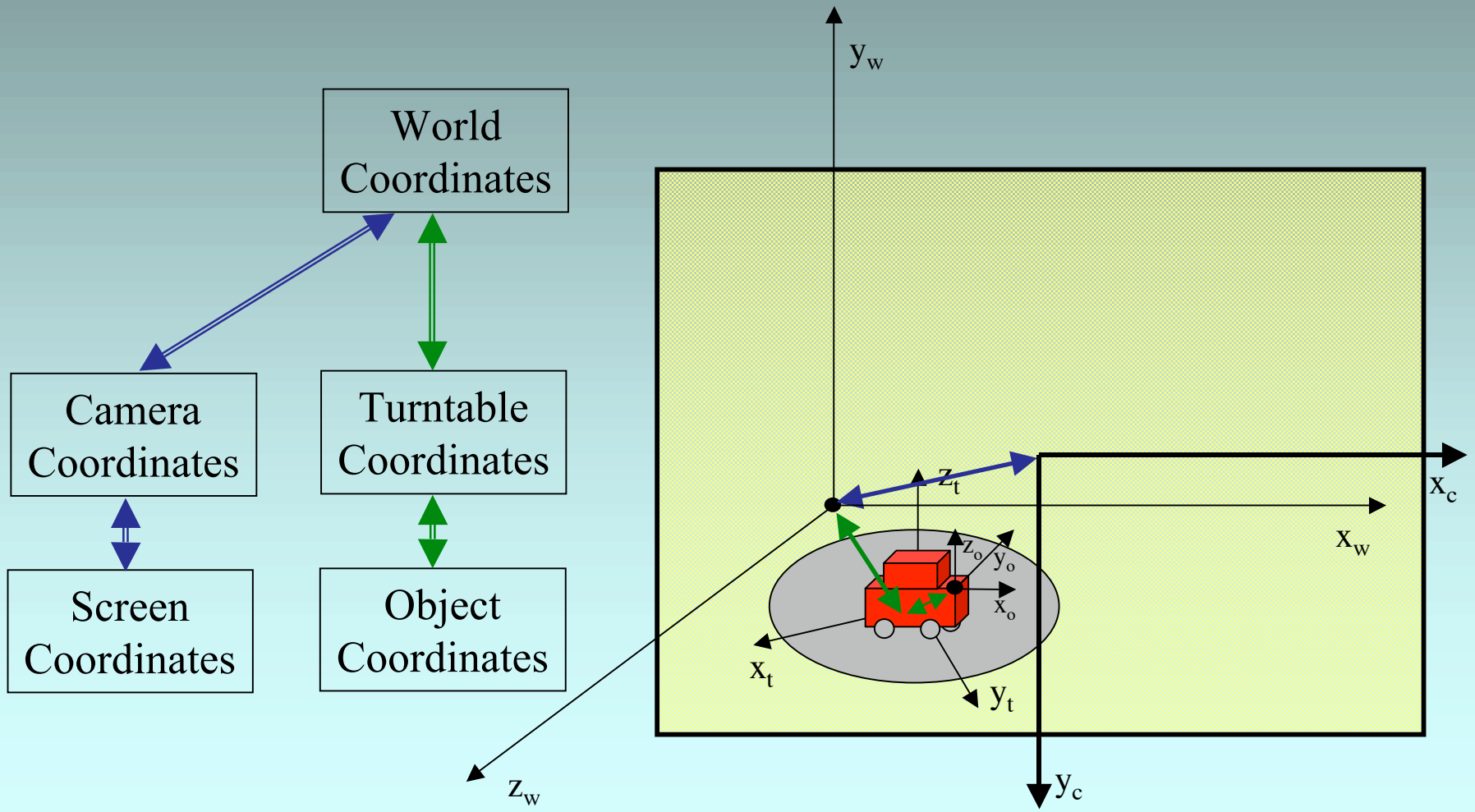
Gudrun Klinker

Jan 20, 2004

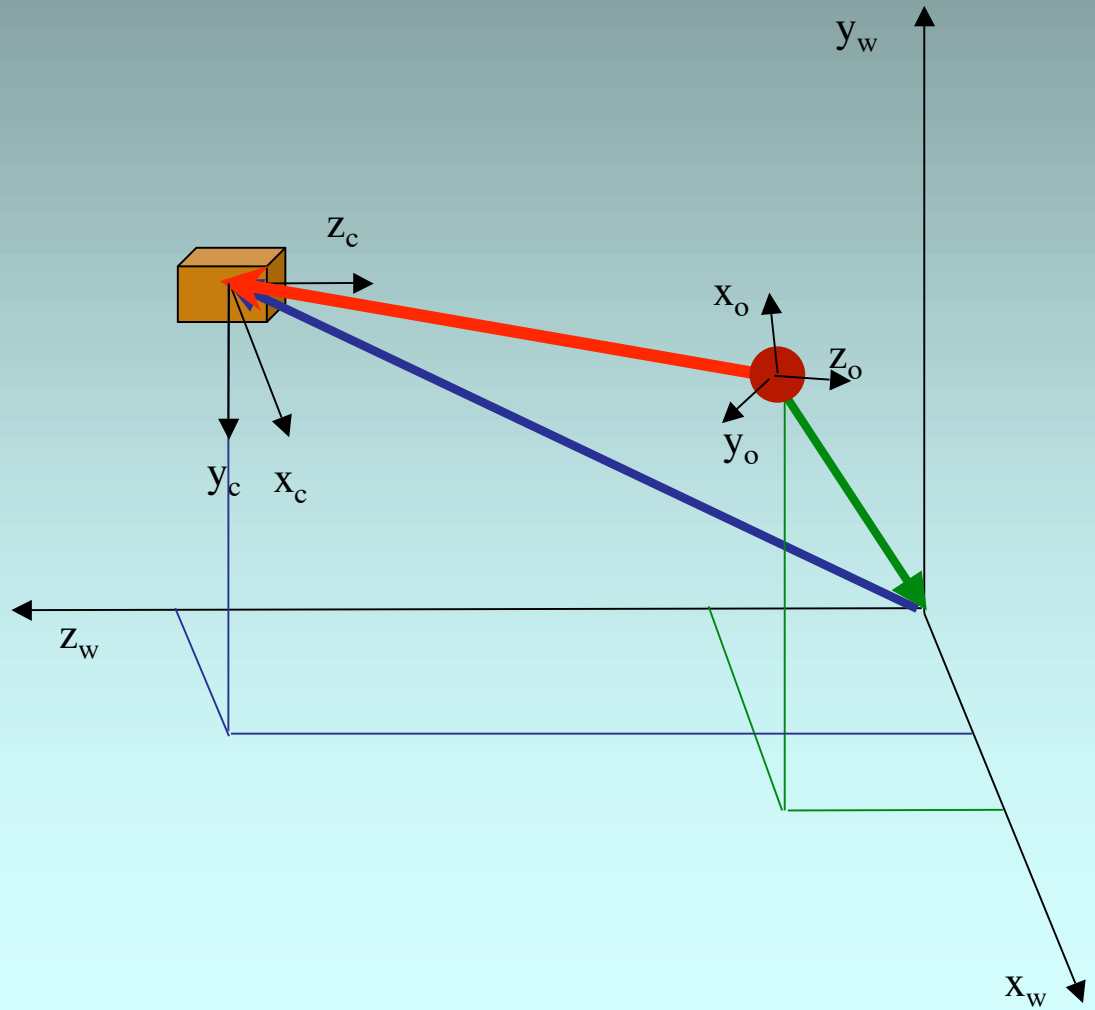
Coordinate Systems in Virtual Worlds



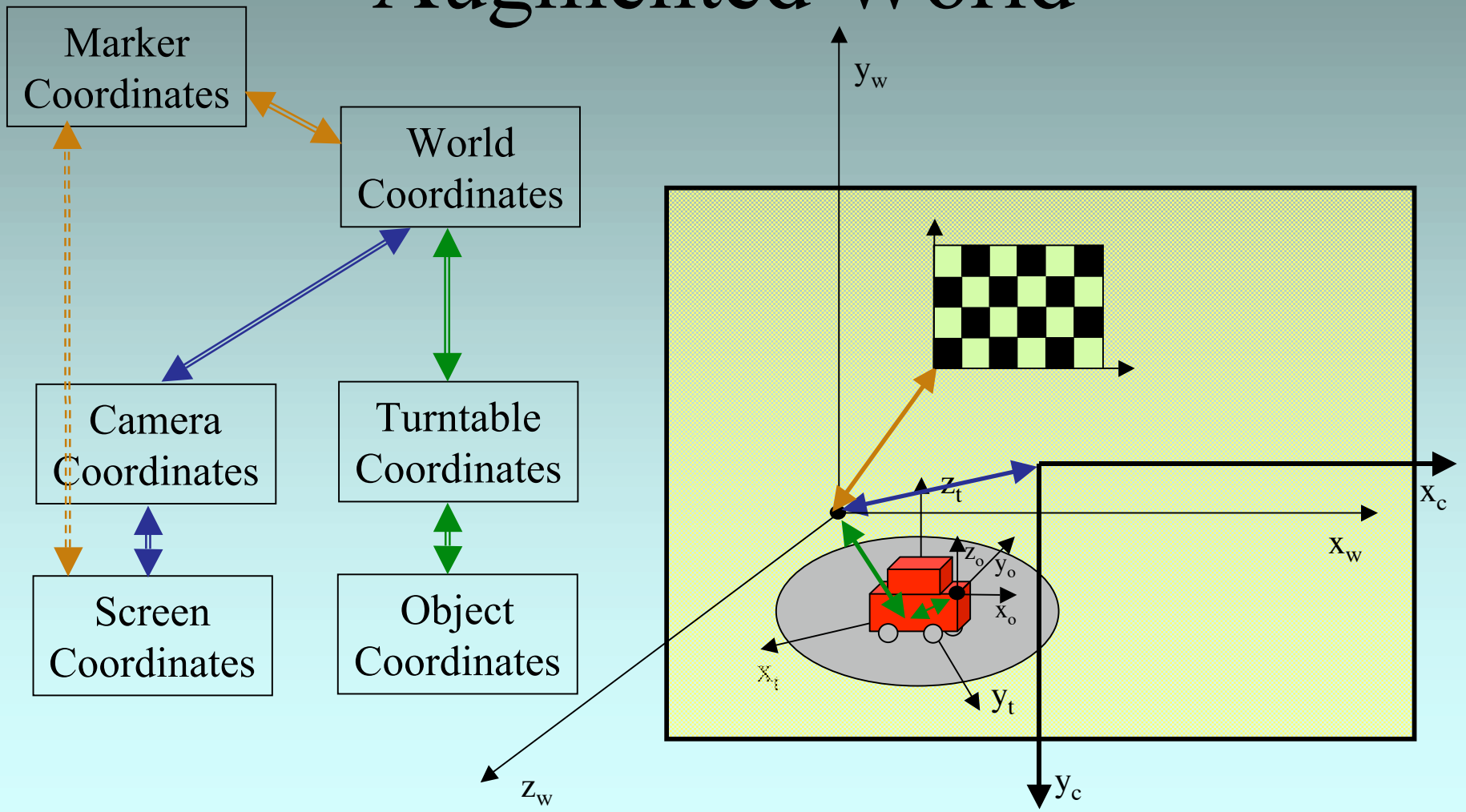
Coordinate Systems Projected onto a Screen



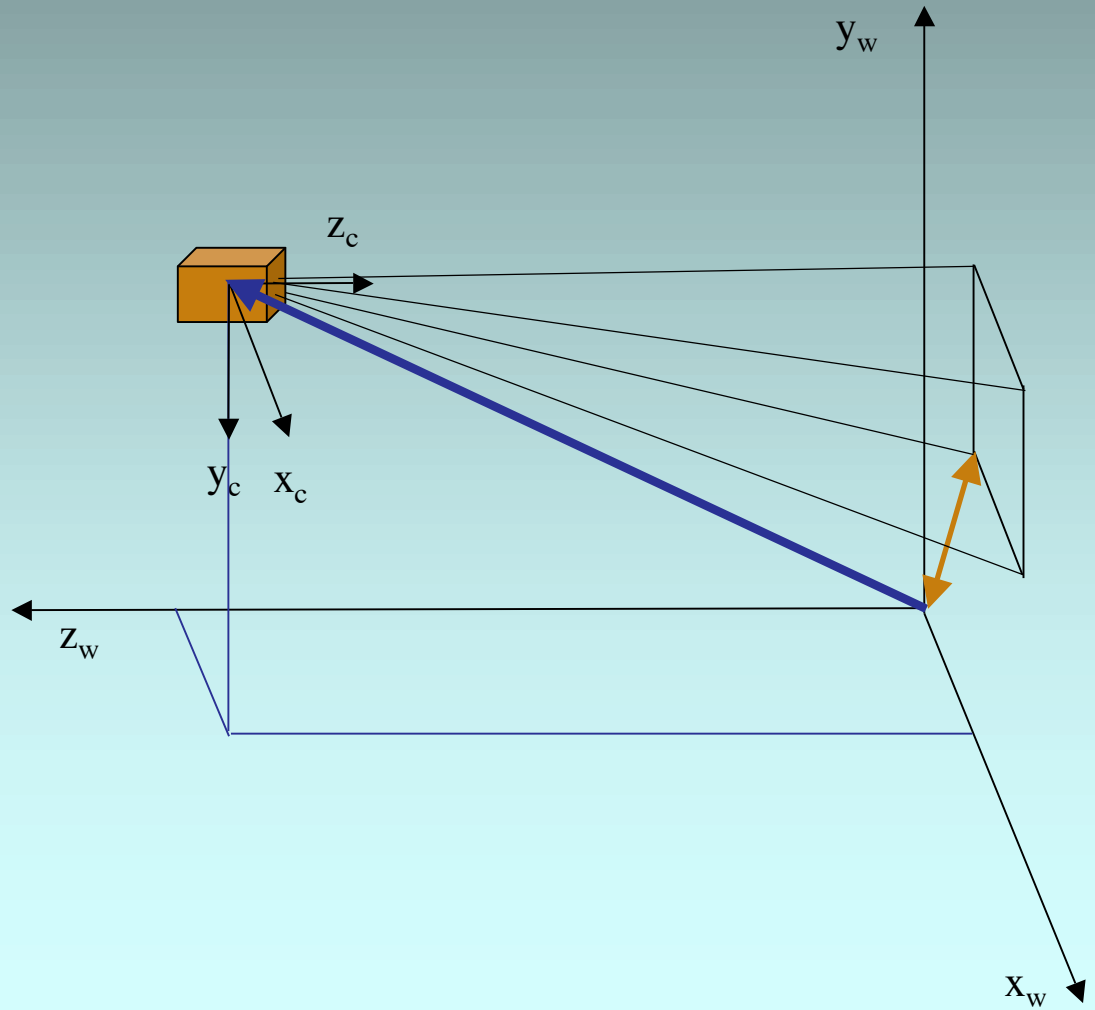
Coordinate Systems Projected onto a Screen



Coordinate Systems in an Augmented World



Coordinate Systems in an Augmented World



Camera Calibration

- Extrinsic parameters (“pose estimation”)
- Intrinsic parameters (“calibration”)

- Tsai’s Approach:
  - Model of image formation steps
  - Two-stage algorithm to determine model parameters

“Classic” Algorithm

- Roger Y. Tsai: A Versatile Camera Calibration Technique for High Accuracy 3D Machine Vision.

  Source code available at

  http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/rgw/www/TsaiCode.html

Tsai's Approach Overview

- World-to-camera transformation: rotation followed by translation
- 6 intrinsic and 6 extrinsic camera parameters
- Parallelism constraint under radial distortion, variable focal length and translation in z
- 2-stage algorithm:
  - Compute 3D orientation, T_x and T_y
  - Compute focal length, distortion coefficients and T_z
  - (other parameters taken from device specifications)
- Separate solutions for coplanar and non-coplanar sets of markers

Image Formation Steps

- Given: Points in world coords (x_w, y_w,z_w)
- World-to-camera transformation. Params: R, T

(x,y,z) ↓ camera coords

- Idealized 3D-to-2D projection. Param: f

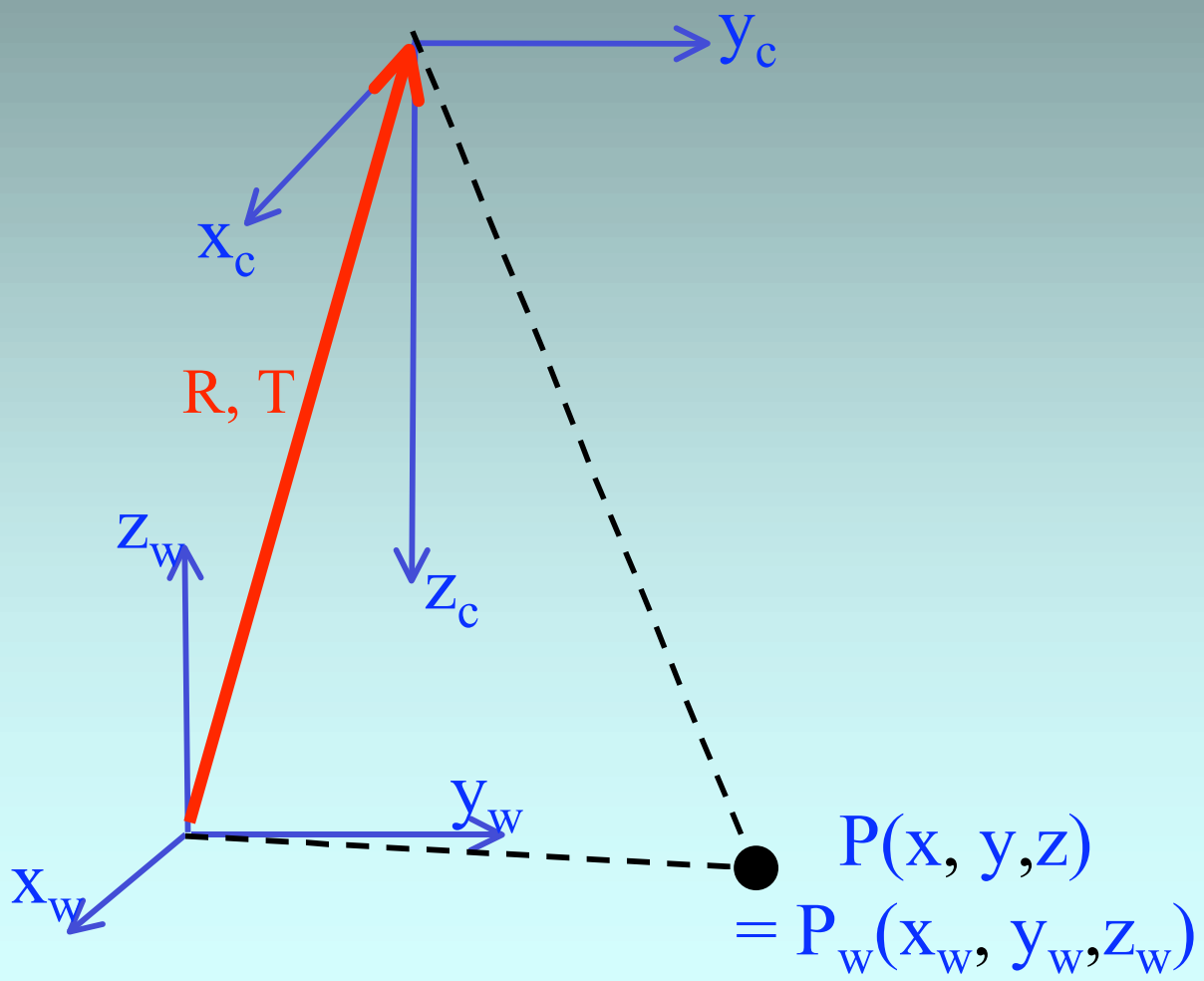
(X_u,Y_u) ↓ ideal image coords

- Radial lens distortion. Params: κ_1, κ_2

(X_d, Y_d) ↓ true image coords

- TV scanning, sampling. Params: s_x, C_x, C_y
- Result: Points in computer image coords (X_f, Y_f)

Step 1:
World-to-Camera Transformation



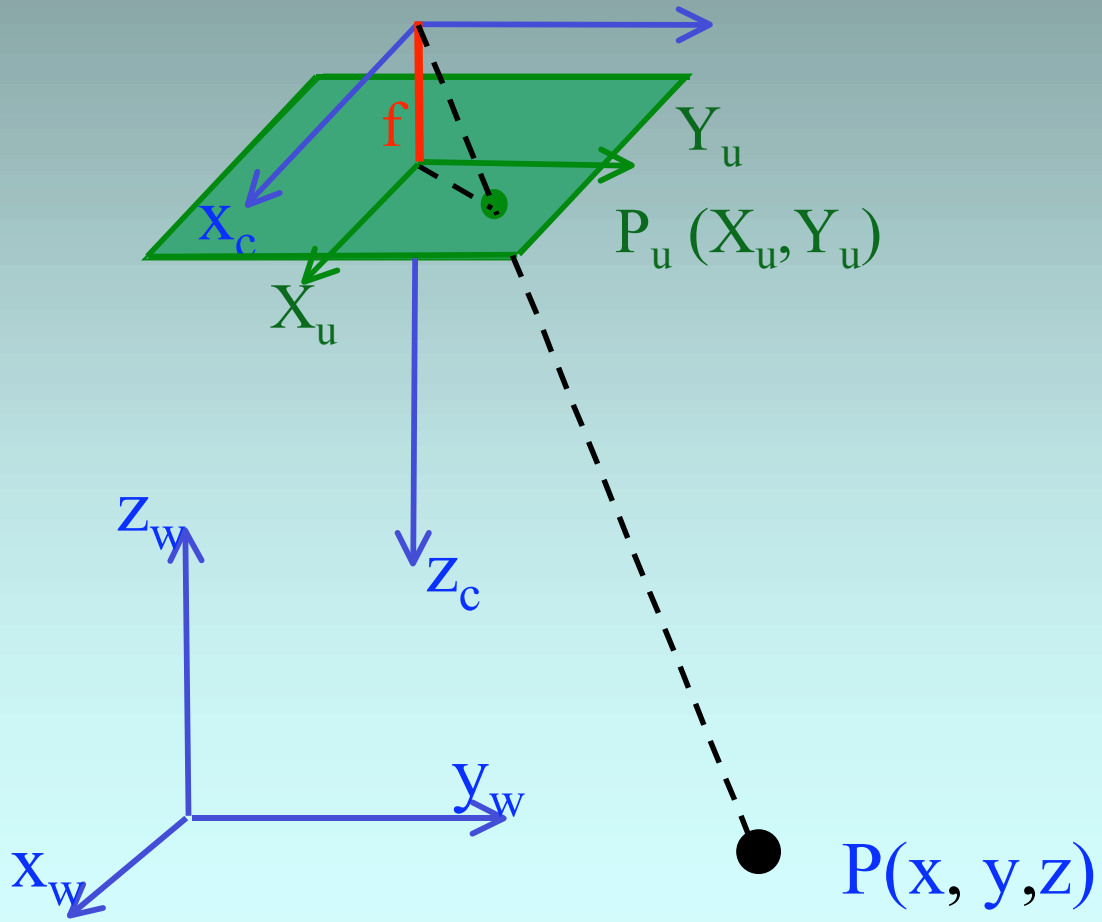
Step 1:
World-to-Camera Transformation

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = R \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} + T$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{pmatrix} \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} + \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}$$

$$\begin{aligned} x &= r_1 x_w + r_2 y_w + r_3 z_w + T_x \\ y &= r_4 x_w + r_5 y_w + r_6 z_w + T_y \\ z &= r_7 x_w + r_8 y_w + r_9 z_w + T_z \end{aligned}$$

Step 2:
Idealized 3D-to-2D Projection



Step 2:
Idealized 3D-to-2D Projection

$$X_u = f \, x/z$$

$$= f \frac{r_1 x_w + r_2 y_w + r_3 z_w + T_x}{r_7 x_w + r_8 y_w + r_9 z_w + T_z}$$

$$Y_u = f \, y/z$$

$$= f \frac{r_4 x_w + r_5 y_w + r_6 z_w + T_y}{r_7 x_w + r_8 y_w + r_9 z_w + T_z}$$

Step 3:
Radial Lens Distortion

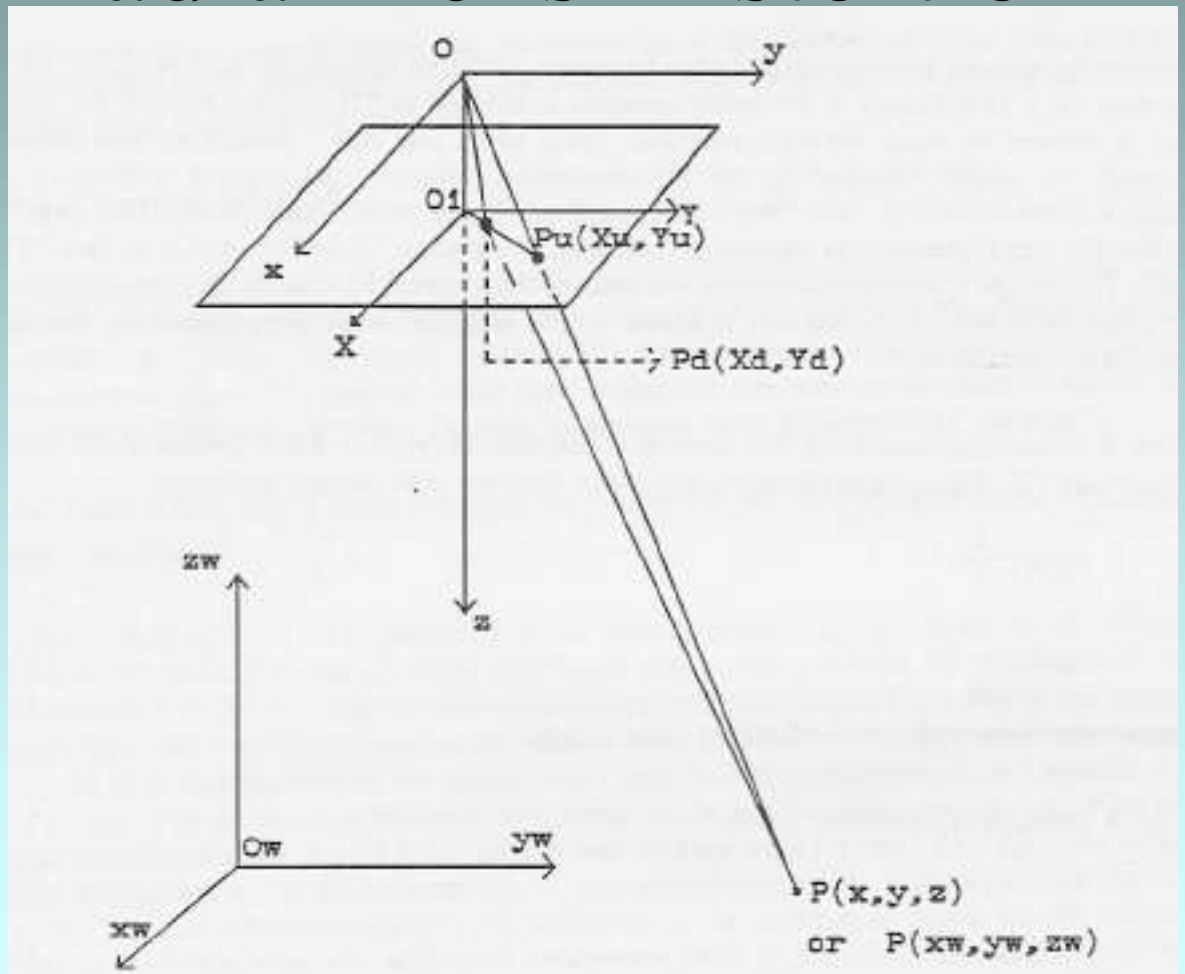


Figure 1. Camera Geometry with Perspective Projection and Radial Lens Distortion

Step 3:
Radial Lens Distortion

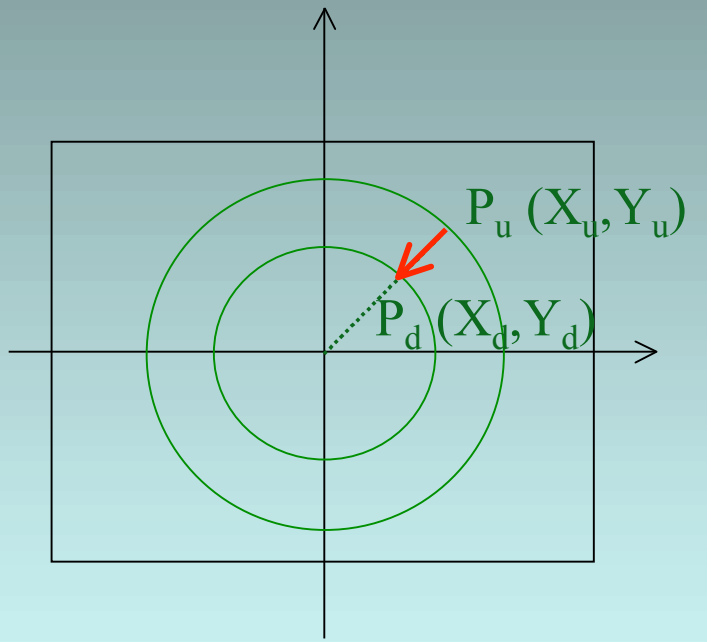
$$X_d = X_u - D_x$$

$$Y_d = Y_u - D_y$$

with:

$$D_x = X_d (\kappa_1 r^2 + \kappa_2 r^4)$$
$$D_y = Y_d (\kappa_1 r^2 + \kappa_2 r^4)$$

$$r = \sqrt{X_d^2 + Y_d^2}$$



Step 4:
TV Scanning, Sampling

$$X_f = (s_x / d_x') X_d + C_x$$

$$Y_f = (1 / d_y) Y_d + C_y$$

with:

- (X_f, Y_f): row, column of image pixel in frame memory
- s_x: uncertainty image scale factor
- (C_x, C_y): center of computer image
- d_x: distance between adjacent sensor elements (x-dir)
- d_y: distance between adjacent sensor elements (y-dir)
- $d_x' = d_x (N_{cx} / N_{fx})$
- N_{cx}: number of sensor elements (x-dir)
- N_{fx}: number of pixels in a line

Image Formation Steps Summary

$$X_u = f\, x/z = f \frac{r_1 x_w + r_2 y_w + r_3 z_w + T_x}{r_7 x_w + r_8 y_w + r_9 z_w + T_z}$$

$$X_u = X_d + D_x = X_d + X_d (\kappa_1 r^2 + \kappa_2 r^4) = (d_x' / s_x)(X_f - C_x) [1 + (\kappa_1 r^2 + \kappa_2 r^4)]$$

$$Y_u = f\, y/z = f \frac{r_4 x_w + r_5 y_w + r_6 z_w + T_y}{r_7 x_w + r_8 y_w + r_9 z_w + T_z}$$

$$Y_u = y_d + D_y = Y_d + Y_d (\kappa_1 r^2 + \kappa_2 r^4) = d_y (Y_f - C_y) [1 + (\kappa_1 r^2 + \kappa_2 r^4)]$$

Two-Stage Camera Calibration Technique

- Preparation:
  - Determine N_{cx}, N_{fx}, d_x' and d_y from device specifications, assume (C_x, C_y) is the center pixel of the image
  - Measure markers (i:1..N) in the scene $(x_w, y_w, z_w)_i$
  - For current image: determine computer image coordinates $(X_f, Y_f)_j$ of all visible markers (j ∈ 1..N)
- Stage 1:
  Compute 3D orientation (R) and position (T_x, T_y) and scale factor (s_x)
- Stage 2:
  Compute effective focal length (f) distortion coefficients (κ_1, κ_2), and z position T_z

Stage 1:
Compute 3D Orientation and Position

- Get distorted image coordinates (X_d, Y_d)
- Compute transformation parameters (5-7 mixed terms of R and T)
- Determine R and T from mixed terms

Illustration

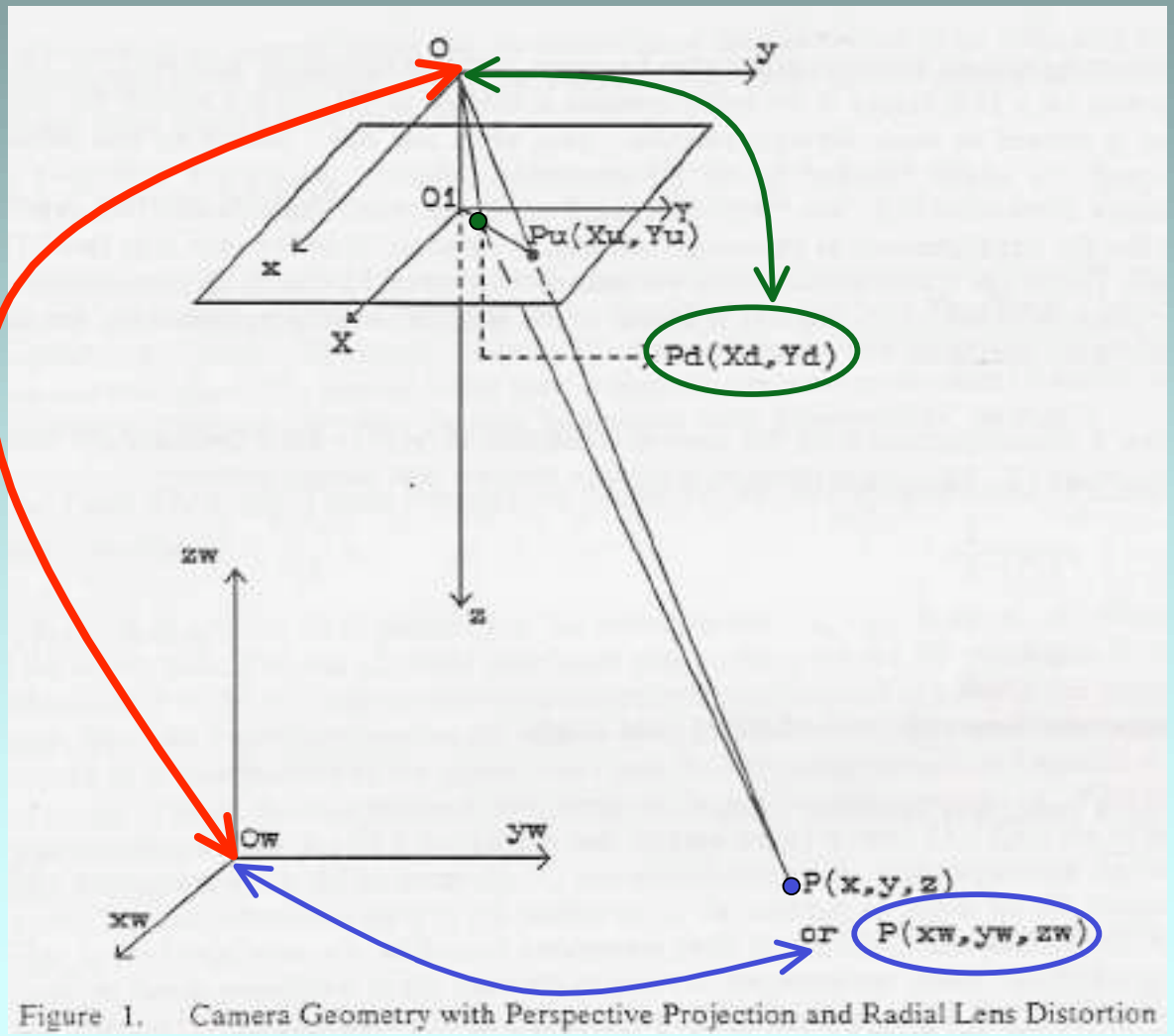
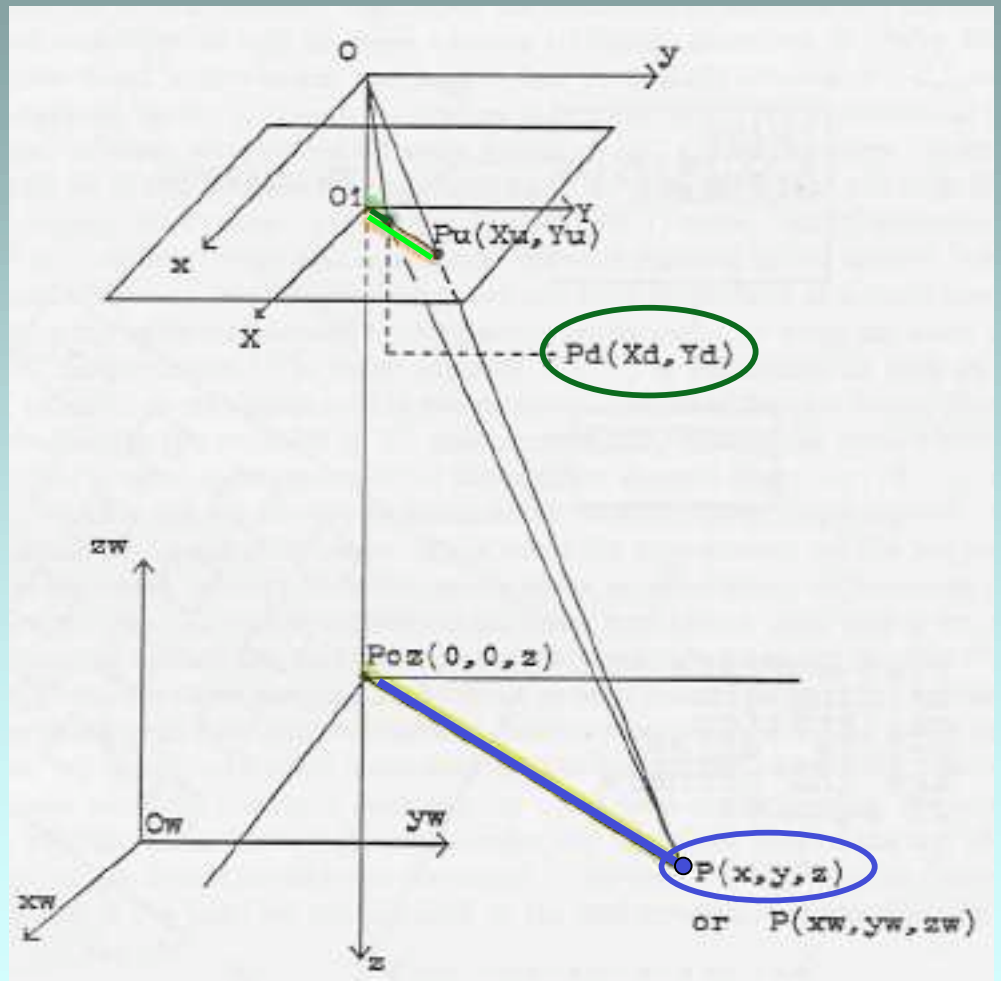


Figure 1. Camera Geometry with Perspective Projection and Radial Lens Distortion

Parallelism Constraint



Radial distortion does not alter the direction of the vector from the origin to the image point:

$\overline{O_f P_u} \parallel \overline{O_f P_d} \parallel \overline{P_{oz} P}$

Parallelism Constraint

Radial distortion does not alter the direction of the vector from the origin to the image point: $\overline{O_f\ P_d} \parallel \overline{P_{oz}\ P}$

$$\begin{pmatrix} 0 \\ 0 \\ f \end{pmatrix} \begin{pmatrix} X_d \\ Y_d \\ f \end{pmatrix} \parallel \begin{pmatrix} 0 \\ 0 \\ z \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$\begin{pmatrix} X_d \\ Y_d \end{pmatrix} = \square \begin{pmatrix} x \\ y \end{pmatrix}$$

$$X_d y = Y_d x$$

Parallelism Constraint

$$X_d\, y = Y_d\, x$$

$$X_d\,(r_4\, x_w + r_5\, y_w + r_6\, z_w + T_y) = Y_d\,(r_1\, x_w + r_2\, y_w + r_3\, z_w + T_x)$$

$$X_d\, T_y = Y_d\,(r_1\, x_w + r_2\, y_w + r_3\, z_w + T_x) - X_d\,(r_4\, x_w + r_5\, y_w + r_6\, z_w)$$

$$X_d = Y_d\, x_w\,(r_1/T_y) + Y_d\, y_w (r_2/T_y) + Y_d\, z_w (r_3/T_y) + Y_d\,(T_x/T_y) - X_d\, x_w\,(r_4/T_y) - X_d\, y_w\,(r_5/T_y) - X_d\, z_w\,(r_6/T_y)$$

Written as vectors and matrix for all markers (i: 1..N):

$$X_{di} = [Y_d\, x_w \quad Y_d\, y_w \quad Y_d\, z_w \quad Y_d \quad -X_d\, x_w \quad -X_d\, y_w \quad -X_d\, z_w]_i \circ [r_1/T_y \quad r_2/T_y \quad r_3/T_y \quad T_x/T_y \quad r_4/T_y \quad r_5/T_y \quad r_6/T_y]$$

Data from each marker becomes a value in a 1xN result vector X and a row in an Nx7 matrix, M, to be multiplied with the 1x7 parameter vector p:

$$\boxed{X = M\, p}$$

Compute Transformation Parameters (Mixed Terms of R and T)

Two cases for using the parallelism constraint:

- Coplanar markers: $z_w = 0$

  $X_d = [Y_d x_w \quad Y_d y_w \quad ~~Y_d z_w~~ \quad Y_d \quad -X_d x_w \quad -X_d y_w \quad ~~-X_d z_w~~]_{i\,o}$

  $[r_1/T_y \quad r_2/T_y \quad ~~r_3/T_y~~ \quad T_x/T_y \quad r_4/T_y \quad r_5/T_y \quad ~~r_6/T_y~~]$

  System of N linear equations with 5 parameters
- Non-coplanar markers

  System of N linear equations with 7 parameters

The Tsai-code uses a routine (lmdif_ from the MINPACK library) that minimizes the square error between predicted and actual marker locations in the image (based on the Levenberg-Marquardt algorithm)

Determine R and T from Mixed Terms (Coplanar Case)

5 Parameters: r_1/T_y, r_2/T_y, T_x/T_y, r_4/T_y, r_5/T_y

$$R: \begin{pmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{pmatrix} \quad \text{submatrix } C: \begin{pmatrix} r_1' & r_2' & . \\ r_4' & r_5' & . \\ . & . & . \end{pmatrix} = \begin{pmatrix} r_1/T_y & r_2/T_y & . \\ r_4/T_y & r_5/T_y & . \\ . & . & . \end{pmatrix}$$

Observations:

- Unit row and column vectors,
- Mutually orthogonal column vectors

$$T_y^2 = \frac{S_r - \sqrt{S_r^2 - 4(r_1' r_5' - r_4' r_2')}}{2\,(r_1' r_5' - r_4' r_2')^2}, \qquad \text{with: } S_r = r_1'^2 + r_2'^2 + r_4'^2 + r_5'^2$$

Determine sign of T_y: try out +/- 3D-to-2D projection for a non-central marker

Determine R and T from Mixed Terms (Coplanar Case)

$$r_1 = r_1' T_y$$
$$r_2 = r_2' T_y$$
$$r_4 = r_4' T_y$$
$$r_5 = r_5' T_y$$
$$T_x = T_x' T_y$$

$$R = \begin{pmatrix} r_1 & r_2 & \sqrt{1 - r_1^2 - r_2^2} \\ r_4 & r_5 & s\sqrt{1 - r_4^2 - r_5^2} \\ r_7 & r_8 & r_9 \end{pmatrix}$$

$$(r_7 \quad r_8 \quad r_9) = (r_1 \quad r_2 \quad r_3) \times (r_4 \quad r_5 \quad r_6)$$

$$s = - \operatorname{sgn} (r_1 r_4 + r_2 r_5)$$

Determine R and T from Mixed Terms (Non-Coplanar Case)

....

Stage 2: Compute Focal Length, Distortion Coefficients, and z Position

- Compute an approximation of f and T_z, ignoring lens distortion

  for every marker i, compute

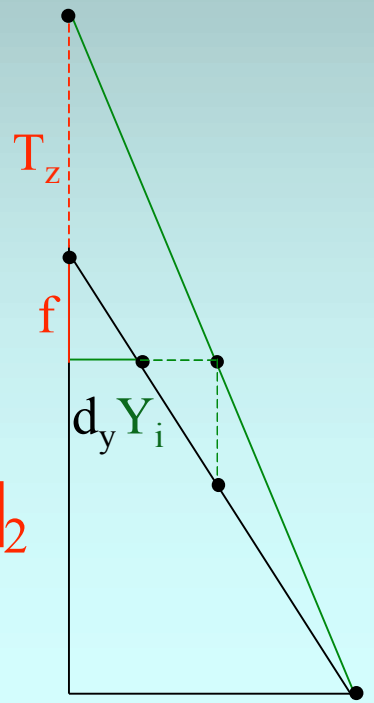
  $$y_i f - T_z d_y Y_i = w_i d_y Y_i$$

  with:

  $$y_i = r_4 x_{wi} + r_5 y_{wi} + r_6 z_{wi} + T_y$$

  $$w_i = r_7 x_{wi} + r_8 y_{wi} + r_9 z_{wi} \qquad = z_i - T_z$$

- Compute exact solution for f, T_z, κ_1 and κ_2



Derivation of formula

for every marker i:

$f y_i / z_i = d_y Y_i$ (3D-to-2d perspective projection + camera sampling)

$f y_i = z_i (d_y Y_i)$

$f y_i - T_z (d_y Y_i) = z_i (d_y Y_i) - T_z (d_y Y_i)$

$f y_i - T_z (d_y Y_i) = (z_i - T_z) (d_y Y_i)$

$f y_i - T_z d_y Y_i = w_i d_y Y_i$

with:

$y_i = r_4 x_{wi} + r_5 y_{wi} + r_6 z_{wi} + T_y$

$w_i = r_7 x_{wi} + r_8 y_{wi} + r_9 z_{wi} = z_i - T_z$

Tsai's Approach Summary

- World-to-camera transformation: rotation followed by translation
- 6 intrinsic and 6 extrinsic camera parameters
- Parallelism constraint under radial distortion, variable focal length and translation in z
- 2-stage algorithm:
  - Compute 3D orientation, T_x and T_y
  - Compute focal length, distortion coefficients and T_z
  - (other parameters taken from device specifications)
- Separate solutions for coplanar and non-coplanar sets of markers

Use of Tsai-Code

- C-program
- How to call the calibration routines
- How to use the results to set up OpenGL

Global Variables

Set of library routines, operating on global variables:

```
extern struct camera_parameters
    { Ncx, Nfx, dx, dy, dpx, dpy, Cx, Cy, sx } cp;
extern struct calibration data
    { point_count, xw[n], yw[n], zw[n], xf [n], yf[n] } cd;
extern struct calibration_constants
    { f, kappa1, p1, p2, Tx, Ty, Tz, Rx, Ry, Rz,
    r1, r2, r3, r4, r5, r6, r7, r8, r9 } cc;
```

Calibration routines

coplanar_calibration ();
coplanar_calibration_with_full_optimization ();
noncoplanar_calibration ();
noncoplanar_calibration_with_full_optimization ();
coplanar_extrinsic_parameter_estimation ();
noncoplanar_extrinsic_parameter_estimation ();

Initialization

example

```
cp.Ncx = w;        cp.Nfx = w;
cp.dx = 0.01;      cp.dy = 0.01;
cp.dpx = 1.0;      cp.dpy = 1.0;
cp.Cx = w/2;       cp.Cy = h/2;
cp.sx = 1.0;

for (i=0; i<n; i++) {
    cd.xw[i] = …;   cd.yw[i] = …;   cd.zw[i] = …;
    cd.Xf[i] = …;   cd.Yf[i] = …;
}
```

Interpretation of Results

t[0] = cc.Tx; t[1] = cc.Ty; t[2] = cc.Tz;

r[0][0] = cc.r1; r[0][1] = cc.r2; r[0][2] = cc.r3;
r[1][0] = cc.r4; r[1][1] = cc.r5; r[1][2] = cc.r6;
r[2][0] = cc.r7; r[2][1] = cc.r6; r[2][2] = cc.r9;

*cx = cc.Cx;
*cy = cc.Cy;
*fx = cc.f * cp.sx;
*fy = cc.f;
*sx = cp.sx;

Use of Results for Open GL

```
dx = (w/2) / fx;                    dy = (h/2) / fy;
alpha_x = 2 * arctan ( dx );        alpha_y = 2 * arctan ( dy );

center_x = 0;                       center_y = 0;
right = dx;                          top = dy;
left = -dx;                          bottom = -dy;
near = 1.0;                         far = 1e6;

glMatrixMode (GL_PROJECTION);
glFrustum (left, right, bottom, top, near, far);
glViewport (0,0,w,h);
```

Use of Results for OpenGL

Translation matrix:
double TMat[16];

1	0	0	0
0	1	0	0
0	0	1	0
tx	-ty	-tz	1

Rotation matrix:
double RMat[16];

r1	-r4	-r7	0
r2	-r5	-r8	0
r3	-r6	-r9	0
0	0	0	1

(OpenGL matrices in column-major order!!! Rotated coordinate system)

```
glMatrixMode (GL_MODELVIEW);
glLoadIdentity ();
glMultMatrixd (TMat);
glMultMatrixd (RMat);
```

(“Fly-Through Mode”)